Mathematische Operatoren:

```
+           // Addition 5 + 6 = 11
-           // Subtraktion 5 - 6 = -1
*           // Multiplikation 5 * 6 = 30
/           // Division 5 / 6 = 0.8333333333333334
%           // Modulo 5 % 6 = 5
//          // Floor-Division 5 // 6 = 0
**          // Exponentiation 5 ** 6 = 15625
```

Vergleichsoperatoren:

```
==          // if 5 == 6:      False (nur Wert wird verglichen)
!=          // if 5 != 6:      True
<           // if 5 < 6:       True
>           // if 5 > 6:       False
<=          // if 5 <= 6:      True
>=          // if 5 >= 6:      False
and         // if 5 < 6 and 8 > 9:       False
or          // if 5 < 6 or 8 > 9:        True
is          // if x is False:            x = 0     False
not         // if not x is False:        True
```

Die if Abfrage:

```
n = 5
x = 10
if n == x:
    print(f"Die Zahl n({n}) ist gleich x({x}).")
elif n < 10:
    print(f"Die Zahl n({n}) ist kleiner als x({x}).")
else:
    print(f"Die Zahl n({n}) ist größer als x({x}).")
```

while Loop:

```
i = 0
while i < 10:
    print(i)
    i += 1
print("Ich habe fertig.")
```

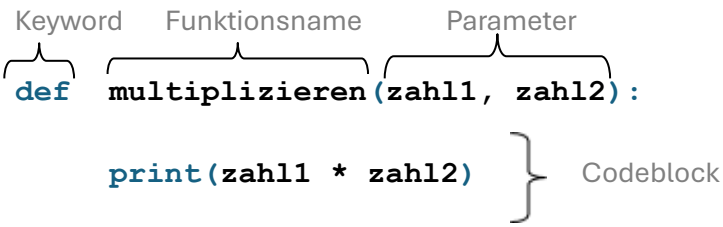
Mit break kann man die Schleife vorzeitig beenden.

Die for Schleife:

```
for i in [2, 4, 6, 8]:

for i in range(10):
```

Funktion ohne Rückgabewert:




```
multiplizieren(5, 6)      // Ergebnis 30 (Funktionsaufruf)
```

Funktion mit Rückgabewert:

```
def  multiplizieren(zahl1, zahl2):

     return zahl1 * zahl2


result = multiplizieren(5, 6)

print(result)       // Ausgabe 30
```

Eine Module in Python Connecting:

```
from myfunctions import *

from myfunctions import dosomething

import myfunction as mf
```

Python "Try Exection Block"

```
# Versuche, die Eingabe in eine Kommazahl umzuwandeln
try:
     stunden_input = stunden_input.replace(',', '.')
     stunden = float(stunden_input)
except:
     print("Ungültige Eingabe! Bitte noch mal versuchen.")
     continue  # Keyword, Codeblock wird wiederholt

# Keywords

ValueError, NameError, TypeError

else:      # gab es keinen Fehler, wird ausgeführt

finally:   # nach try, wird immer ausgeführt
```

Kommastellen anzeigen

```
print(f"\nGesamtbetrag: {gesamtbetrag:.2f} €")
```